# Spring Framework(STS3)

<스프링 툴 다운로드>
https://spring.io/tools
spring-tool-suite-3.9.18.RELEASE-e4.21.0-win32-x86_64 다운로드

톰캣 9.0 -> 불러오기
<톰캣 web.xml 포트 변경> - 이클립스랑 포트 충돌되는 것 방지

JRE11 버전으로 호환시켜주기

<스프링 프레임워크에서 서블릿 연결하기 - web.xml>

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd" id="WebApp_ID" version="4.0">
  <display-name>SpringProject01</display-name>
  <servlet>
      <servlet-name>springapp</servlet-name>
      <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  </servlet>
  <servlet-mapping>
      <servlet-name>springapp</servlet-name>
      <url-pattern>*.sp</url-pattern>
  </servlet-mapping>
  <filter>
            <filter-name>encodingFilter</filter-name>

<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
            <init-param>
                  <param-name>encoding</param-name>
                  <param-value>utf-8</param-value>
            </init-param>
  </filter>
  <filter-mapping>
      <filter-name>encodingFilter</filter-name>
      <url-pattern>/*</url-pattern>
  </filter-mapping>

  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

**<스프링 빈 지정 springapp-servlet.xml(서블릿 형식)>**

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

<!-- test -->
<bean name="/test.sp" class="com.test.TestController"></bean>

</beans>
```

**<자바 파일 설정 - 스프링 빈과 연동하여 MVC 설정(서블릿 형식)>**

```
package com.test;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.AbstractController;

public class TestController extends AbstractController{

	@Override
	protected ModelAndView handleRequestInternal(HttpServletRequest req,
HttpServletResponse resp) throws Exception {

		return new ModelAndView("WEB-INF/jsp/result.jsp");
	}

}
```

**<MySQL Workbench 설치 - DB 활용>**

**<나머지는 소스파일 참조>**

**기존 JSP Servlet에서는 스태틱으로 만들었기 때문에 싱글톤으로 만든 것이기 때문에 내용을 수정하려면 여러 번 수정을 거쳐야 함.**
**그렇기에 외부로 빼서 하면 유지보수가 편함.**
**DAO객체를 매번 받을 필요가 없는 것이 스프링과 서블릿의 차이점.**
**스프링에서는 생성자, getter와 setter, @(어노테이션)을 사용하여 주입을 할 수 있음.**

**<어노테이션(@) 사용 - 유기적인 유지보수>**
**@Repository, @Controller, @AutoWired를 사용**

**@Repository:**
**@Repository 애노테이션은 스프링에서 데이터 액세스 계층의 구성 요소를 나타냅니다.**
**주로 DAO(Data Access Object) 클래스에 적용됩니다.**
**데이터베이스와의 상호 작용, 데이터 검색 및 저장, 예외 처리 등 데이터 액세스 작업을 수행하는 클래스를 표시하는 데 사용됩니다.**

@Repository 애노테이션이 지정된 클래스는 스프링에 의해 자동으로 인스턴스화되고, 예외 변환 기능과 같은 데이터 액세스 관련 기능을 제공하는 빈으로 등록됩니다.

@Controller:
@Controller 애노테이션은 스프링 MVC 애플리케이션에서 웹 요청을 처리하는 컨트롤러 클래스를 나타냅니다.
주로 사용자의 요청을 처리하고 응답을 반환하는 역할을 수행합니다.
MVC 패턴에서 컨트롤러는 사용자의 입력을 받고 모델을 업데이트한 후 적절한 뷰를 선택하여 응답을 생성합니다.
@Controller 애노테이션이 지정된 클래스는 스프링에 의해 자동으로 인스턴스화되고, 요청 매핑 및 요청 처리를 수행하는 빈으로 등록됩니다.

@Autowired:
@Autowired 애노테이션은 스프링에서 의존성 주입(Dependency Injection)을 수행하는 데 사용됩니다.
의존성 주입은 객체 간의 결합을 완화시키고 관리를 용이하게 하는 데 도움을 줍니다.
@Autowired 애노테이션은 생성자, 필드, 메서드에 적용될 수 있으며, 스프링은 해당 타입에 해당하는 빈을 찾아서 자동으로 주입합니다.
주로 의존하는 객체를 자동으로 주입받아 사용할 때 사용됩니다.

Spring Boot - xml의 역할을 자바로 변환시켜서 만들어 줌.

---

@(어노테이션)을 활용하여 스프링에서 찾아서 구동할 수 있도록 빈에서 추가를 해준다.
```
<context:component-scan base-package="com.test2.controller"/>
```

Legacy Project - 자바 구버전이니 자바 버전 세팅해줘야함.
Maven Project

View - Controller - Service(Transaction 처리용도) - Repository(DAO) - DB

<MyBatis 사용환경 설정>
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/maven-v4_0_0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<groupId>com.myspring</groupId>
	<artifactId>app04</artifactId>
	<name>SpringProject04_MyBatis</name>
	<packaging>war</packaging>
	<version>1.0.0-BUILD-SNAPSHOT</version>
	<properties>
		<java-version>11.0</java-version>
		<org.springframework-version>5.2.9.RELEASE</org.springframework-version>
		<org.aspectj-version>1.6.10</org.aspectj-version>
		<org.slf4j-version>1.6.6</org.slf4j-version>
```

```
</properties>
<dependencies>
	<!-- Spring -->
	<dependency>
		<groupId>org.springframework</groupId>
		<artifactId>spring-context</artifactId>
		<version>${org.springframework-version}</version>
		<exclusions>
			<!-- Exclude Commons Logging in favor of SLF4j -->
			<exclusion>
				<groupId>commons-logging</groupId>
				<artifactId>commons-logging</artifactId>
			</exclusion>
		</exclusions>
	</dependency>
	<dependency>
		<groupId>org.springframework</groupId>
		<artifactId>spring-webmvc</artifactId>
		<version>${org.springframework-version}</version>
	</dependency>

	<!-- JDBC -->
	<!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
	<dependency>
		<groupId>org.springframework</groupId>
		<artifactId>spring-jdbc</artifactId>
		<version>5.2.9.RELEASE</version>
	</dependency>

	<dependency>
		<groupId>org.springframework</groupId>
		<artifactId>spring-jdbc</artifactId>
		<version>${org.springframework-version}</version>
	</dependency>

	<!-- Mysql -->
	<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
	<dependency>
		<groupId>mysql</groupId>
		<artifactId>mysql-connector-java</artifactId>
		<version>8.0.28</version>
	</dependency>

	<!-- mybatis -->
	<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis -->
	<dependency>
		<groupId>org.mybatis</groupId>
		<artifactId>mybatis</artifactId>
		<version>3.5.6</version>
	</dependency>
```

```
<!-- mybatis Spring -->
<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring -->
<dependency>
	<groupId>org.mybatis</groupId>
	<artifactId>mybatis-spring</artifactId>
	<version>1.3.2</version>
</dependency>

<!-- AspectJ -->
<dependency>
	<groupId>org.aspectj</groupId>
	<artifactId>aspectjrt</artifactId>
	<version>${org.aspectj-version}</version>
</dependency>

<!-- Logging -->
<dependency>
	<groupId>org.slf4j</groupId>
	<artifactId>slf4j-api</artifactId>
	<version>${org.slf4j-version}</version>
</dependency>
<dependency>
	<groupId>org.slf4j</groupId>
	<artifactId>jcl-over-slf4j</artifactId>
	<version>${org.slf4j-version}</version>
	<scope>runtime</scope>
</dependency>
<dependency>
	<groupId>org.slf4j</groupId>
	<artifactId>slf4j-log4j12</artifactId>
	<version>${org.slf4j-version}</version>
	<scope>runtime</scope>
</dependency>
<dependency>
	<groupId>log4j</groupId>
	<artifactId>log4j</artifactId>
	<version>1.2.15</version>
	<exclusions>
		<exclusion>
			<groupId>javax.mail</groupId>
			<artifactId>mail</artifactId>
		</exclusion>
		<exclusion>
			<groupId>javax.jms</groupId>
			<artifactId>jms</artifactId>
		</exclusion>
		<exclusion>
			<groupId>com.sun.jdmk</groupId>
			<artifactId>jmxtools</artifactId>
```

```
                </exclusion>
                <exclusion>
                    <groupId>com.sun.jmx</groupId>
                    <artifactId>jmxri</artifactId>
                </exclusion>
            </exclusions>
            <scope>runtime</scope>
        </dependency>

        <!-- @Inject -->
        <dependency>
            <groupId>javax.inject</groupId>
            <artifactId>javax.inject</artifactId>
            <version>1</version>
        </dependency>

        <!-- Servlet -->
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>servlet-api</artifactId>
            <version>2.5</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>javax.servlet.jsp</groupId>
            <artifactId>jsp-api</artifactId>
            <version>2.1</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jstl</artifactId>
            <version>1.2</version>
        </dependency>

        <!-- Test -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.7</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <artifactId>maven-eclipse-plugin</artifactId>
                <version>2.9</version>
                <configuration>
                    <additionalProjectnatures>
```

```
<projectnature>org.springframework.ide.eclipse.core.springnature</projectnature>
                                        </additionalProjectnatures>
                                        <additionalBuildcommands>

<buildcommand>org.springframework.ide.eclipse.core.springbuilder</buildcommand>
                                        </additionalBuildcommands>
                                        <downloadSources>true</downloadSources>
                                        <downloadJavadocs>true</downloadJavadocs>
                                </configuration>
                        </plugin>
                        <plugin>
                                <groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-compiler-plugin</artifactId>
                                <version>2.5.1</version>
                                <configuration>
                                        <source>1.6</source>
                                        <target>1.6</target>
                                        <compilerArgument>-Xlint:all</compilerArgument>
                                        <showWarnings>true</showWarnings>
                                        <showDeprecation>true</showDeprecation>
                                </configuration>
                        </plugin>
                        <plugin>
                                <groupId>org.codehaus.mojo</groupId>
                                <artifactId>exec-maven-plugin</artifactId>
                                <version>1.2.1</version>
                                <configuration>
                                        <mainClass>org.test.int1.Main</mainClass>
                                </configuration>
                        </plugin>
                </plugins>
        </build>
</project>

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
 PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
 "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
 <mappers>
 <mapper resource="Person.xml"/>
 </mappers>
</configuration>

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
 PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
 "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
 <mapper namespace="com.myspring.Person">
```

```
</mapper>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.springframework.org/schema/beans
https://www.springframework.org/schema/beans/spring-beans.xsd">

	<!-- Root Context: defines shared resources visible to all other web components
-->
	<bean id="ds"
		class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
		<property name="driverClass" value="com.mysql.cj.jdbc.Driver" />
		<property name="url"
value="jdbc:mysql://localhost:3306/springdb?useSSL=false&amp;serverTimezone=Asia/Seoul&a
mp;characterEncoding=UTF-8" />
		<property name="username" value="root" />
		<property name="password" value="root" />
	</bean>

	<!-- mybatis -->
	<bean id="sqlSessionFactoryBean"
	class="org.mybatis.spring.SqlSessionFactoryBean">
	<property name="dataSource" ref="ds"/>
	<property name="configLocation" value="classpath:/Configuration.xml"/>
	</bean>

	<bean id="sqlMapper" class="org.mybatis.spring.SqlSessionTemplate">
	<constructor-arg index="0" ref="sqlSessionFactoryBean"/>
	</bean>
</beans>
```

나머지는 소스코드 파일 참조

---

2023.6.26. Spring Framework

<Dynamic SQL - MyBatis>

```
	<sql id="search">
	<where>
		<if test="word!=null field='name'">
			name like'%${word}%'
		</if>
		<if test="word!=null field='job'">
			job like'%${word}%'
		</if>
		</where>
	</sql>
```

<CONCAT(문자연결) 활용>

```
<sql id="search">
<where>
        <if test="word!=null field='name'">
                name like'%${word}%'
        </if>
        <if test="word!=null field='job'">
                job like CONCAT('%',#{word},'%')
        </if>
        </where>
</sql>
```

Maven Repository 사이트 - HikariCP 4.0.3 - pom.xml 붙여넣기
Project Lombok
lombok 1.18.24 클릭-specify location 클릭(이클립스 체크 해제)-STS와 연결-설치 및 재실행

---

<@Select>
@Select는 MyBatis 프레임워크에서 사용되는 애노테이션입니다. MyBatis는 SQL 매핑을 위한 프레임워크로서, @Select 애노테이션을 사용하여 데이터베이스에서 데이터를 조회하는 SQL 문을 지정할 수 있습니다.

<beforeSend function>
beforeSend는 jQuery의 AJAX 요청 중에 호출되는 콜백 함수입니다. AJAX 요청이 서버로 전송되기 전에 실행되며, 주로 요청 전에 필요한 설정이나 준비 작업을 수행하는 데 사용됩니다.

beforeSend 콜백 함수는 AJAX 요청 객체(XMLHttpRequest)를 매개변수로 받아 사용할 수 있습니다. 이를 통해 요청 헤더(header)를 수정하거나 요청 데이터(data)를 조작하는 등의 작업을 수행할 수 있습니다.

@RequestBody는 스프링 프레임워크에서 컨트롤러의 메서드 파라미터에 사용되는 애노테이션입니다. 이 애노테이션은 HTTP 요청의 본문(body) 데이터를 해당 메서드 파라미터에 자동으로 바인딩해줍니다.
JSON형태를 처리하여서 보여줄 때 사용해야 함.

Maven Repository - Jackson Databind(JSON, GSON보다 더 좋음) - 2.15.0 스크립트 복사 pom.xml에 넣어주기

@ResponseBody는 스프링 프레임워크에서 컨트롤러 메서드의 반환 값을 HTTP 응답 본문으로 사용하도록 지정하는 애노테이션입니다. 이 애노테이션을 사용하면 스프링은 메서드의 반환 값이 HTTP 응답 본문에 직접 쓰여지도록 처리합니다.

일반적으로, 스프링 MVC에서 컨트롤러 메서드는 뷰를 반환하거나 데이터를 모델에 담아서 뷰로 전달하는 용도로 사용됩니다. 이때 @ResponseBody 애노테이션을 사용하면 메서드의 반환 값이 뷰를 통하지 않고 직접 HTTP 응답 본문에 작성됩니다.

return "success"하면 success.jsp로 넘어감
@ResponseBody 애노테이션을 사용하여 Jackson Databind가 return 유형을 스트링으로 바꿔서

처리함.

<@AllArgsConstructor>
@AllArgsConstructor는 Lombok 라이브러리에서 제공하는 애노테이션입니다. 이 애노테이션은 클래스의 모든 필드를 사용하는 생성자를 자동으로 생성해줍니다.

일반적으로 Java 클래스를 작성할 때, 생성자를 정의하고 필드에 대한 매개변수를 전달하는 작업을 수동으로 처리해야 합니다. 하지만 @AllArgsConstructor 애노테이션을 사용하면 이러한 번거로운 작업을 자동화할 수 있습니다.

@AllArgsConstructor 애노테이션을 클래스에 적용하면, 해당 클래스의 모든 필드를 사용하는 생성자가 자동으로 생성됩니다. 이 생성자는 클래스의 모든 필드를 매개변수로 받아들이고, 해당 필드에 값을 할당합니다.

<RESTful API>
<RESTful에 대한 이해>
RESTful은 Representational State Transfer의 약어로, 웹 서비스 아키텍처 디자인의 원칙을 나타냅니다. RESTful은 자원(Resource)을 URI(Uniform Resource Identifier)로 표현하고, HTTP 프로토콜을 사용하여 자원에 대한 조작을 수행하는 방식을 의미합니다.

RESTful 아키텍처의 주요 특징은 다음과 같습니다:

자원 중심 (Resource-Oriented): RESTful은 자원을 중심으로 설계됩니다. 각 자원은 고유한 식별자인 URI를 가지며, URI를 통해 자원에 접근하고 조작할 수 있습니다. 예를 들어, /users는 사용자 자원을 나타내는 URI입니다.

HTTP 메서드 활용: RESTful은 HTTP 프로토콜의 메서드를 활용하여 자원에 대한 조작을 수행합니다. 주요 HTTP 메서드인 GET, POST, PUT, DELETE를 사용하여 자원의 조회, 생성, 수정, 삭제를 표현합니다.

상태를 전이하지 않음 (Stateless): RESTful은 상태를 전이하지 않는 아키텍처입니다. 서버는 각 요청을 독립적으로 처리하며, 클라이언트의 상태를 유지하지 않습니다. 이는 서버의 확장성과 클라이언트의 자율성을 높이는데 도움을 줍니다.

자기 설명적 메시지 (Self-descriptive messages): RESTful은 메시지 자체가 어떤 동작을 수행하는지 명확하게 설명할 수 있어야 합니다. HTTP의 표준 메시지 형식인 MIME 타입을 활용하여 메시지를 전달하고, 추가적인 정보를 헤더에 포함하여 자기 설명적인 특성을 갖습니다.

계층 구조 (Layered System): RESTful은 계층 구조를 가질 수 있습니다. 클라이언트와 서버 사이에 중간 계층을 두어 확장성, 보안, 로드 밸런싱 등의 기능을 제공할 수 있습니다.

RESTful 아키텍처는 웹 서비스의 디자인을 단순화하고 확장 가능한 시스템을 구축하는데 도움을 줍니다. 이를 따르면 URI를 통해 자원을 표현하고, HTTP 메서드를 활용하여 자원에 대한 조작을 수행할 수 있습니다.

@PathVariable은 Spring Framework에서 제공하는 애노테이션 중 하나로, RESTful 웹 애플리케이션에서 URI 경로에 포함된 변수 값을 추출하는 데 사용됩니다.
@PathVariable로 경로를 지정해 줌.

<연-월-일 순으로 날짜 나타내기>

```
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"  %>
<td><fmt:formatDate value="${board.regdate}" pattern="yyyy-MM-dd"/></td>
```

---

<@RestController>
@RestController는 스프링 프레임워크에서 제공하는 어노테이션 중 하나로, RESTful 웹 서비스를 구현하기 위해 사용됩니다. @RestController 어노테이션을 클래스에 적용하면 해당 클래스는 RESTful 웹 서비스의 엔드포인트로 동작하게 됩니다.

@RestController 어노테이션은 @Controller와 @ResponseBody 어노테이션을 합친 것과 동일한 역할을 수행합니다. 즉, @Controller 어노테이션을 사용하여 컨트롤러 클래스를 정의하여, 메서드에 @ResponseBody 어노테이션을 사용하여 HTTP 응답 본문으로 반환되는 데이터를 직접 응답으로 변환하는 작업을 수행할 필요가 없어집니다.

@RestController를 사용하여 웹 서비스를 개발할 때, 자동으로 JSON 또는 XML 변환과 HTTP 응답 헤더 설정 등의 작업을 처리해주기 때문에 간편하게 RESTful API를 개발할 수 있습니다.

<@Transactional>
@Transactional은 스프링 프레임워크에서 제공하는 어노테이션으로, 데이터베이스 트랜잭션을 관리하기 위해 사용됩니다. @Transactional 어노테이션을 메서드나 클래스에 적용하면 해당 메서드 또는 클래스 내에서 실행되는 데이터베이스 관련 작업들이 하나의 트랜잭션으로 묶이게 됩니다.

@Transactional 어노테이션을 사용하면 다음과 같은 이점을 얻을 수 있습니다:

자동으로 트랜잭션 관리: @Transactional 어노테이션이 적용된 메서드가 호출되면, 스프링은 자동으로 데이터베이스 트랜잭션을 시작하고, 메서드 실행이 성공하면 트랜잭션을 커밋하고, 예외가 발생하면 트랜잭션을 롤백합니다.

선언적 트랜잭션 설정: @Transactional 어노테이션을 사용하여 트랜잭션 관련 설정을 간단하게 선언적으로 처리할 수 있습니다. 예를 들어, 트랜잭션의 격리 수준, 읽기 전용 트랜잭션 설정, 롤백 조건 등을 어노테이션 속성으로 지정할 수 있습니다.

예외 처리와 롤백: @Transactional 어노테이션은 기본적으로 RuntimeException 및 그 하위 예외들에 대해서만 롤백을 수행합니다. 따라서, 예외가 발생하면 자동으로 트랜잭션을 롤백시키고, 예외 처리를 위해 별도의 try-catch 문을 작성할 필요가 없습니다.

@Transactional 어노테이션을 사용하려면 스프링의 트랜잭션 관리 기능을 설정해야 합니다. 이를 위해 스프링 구성 파일에 @EnableTransactionManagement 어노테이션을 추가하거나 XML 설정 파일에서 <tx:annotation-driven> 요소를 사용하여 트랜잭션 관리 기능을 활성화해야 합니다.

<FileUpload>
Maven Repository - Apache Commons FileUpload 1.3.3 소스 붙여 넣기

---

<Spring FrameWork 특징>
경량 컨테이너
- 자바 객체(Spring Bean)를 다룬다.

POJO - Plain_Old_Java_Object

IoC - Inversion of Control 제어의 역전
- 제어가 사용자가 아닌 Spring이 컨트롤

IoC 구현을 DI(Dependency Injection)를 통해서 주입
DI : 의존성 주입

AOP(Aspect-Oriented Programming, AOP) : 관점 지향 프로그래밍
공통적으로 사용하는 기능을 분리하여 관리
보안, 로깅, Transaction

@Log, @Aspect, @Component은 Spring Framework에서 사용되는 어노테이션입니다.

@Log: @Log 어노테이션은 lombok 라이브러리에서 제공하는 어노테이션으로, 자동으로 로깅 코드를 생성해주는 기능을 제공합니다. 주로 클래스 또는 메서드에 적용하여 로깅을 간편하게 구현할 수 있습니다.

@Aspect: @Aspect 어노테이션은 AOP (관점 지향 프로그래밍)에서 사용되는 어노테이션입니다. @Aspect 어노테이션이 적용된 클래스는 공통 기능인 "Aspect"를 정의하고, 이를 다른 클래스의 특정 지점에 적용할 수 있습니다. 예를 들어, 트랜잭션 관리, 보안 처리 등과 같은 공통 기능을 Aspect로 정의하여 여러 클래스에 적용할 수 있습니다.

@Component: @Component 어노테이션은 Spring Framework의 컴포넌트 스캔을 통해 빈으로 등록될 클래스임을 나타내는 어노테이션입니다. @Component 어노테이션을 사용하면 해당 클래스가 Spring의 ApplicationContext에서 관리되는 빈으로 등록됩니다. @Component 어노테이션은 일반적인 스프링 빈으로 사용되며, @Controller, @Service, @Repository 등의 세부적인 어노테이션들은 @Component 어노테이션을 확장한 것입니다.

이러한 어노테이션들은 Spring Framework에서 제공하는 다양한 기능을 활용하고 관리하기 위해 사용되며, 애플리케이션의 로깅, AOP, 빈 등록과 같은 측면에서 개발 생산성을 향상시키는데 도움을 줍니다.

Maven Repository - AspectJ Weaver 1.9.19 소스 복사

<eGov>
전자정부 사이트 - 개발자 교육 - 교육자료

<비밀번호 보안 구축>
Maven Repository
Spring Security Core 5.4.2
Spring Security Web 5.4.2
Spring Security Config 5.4.2
Spring security-taglibs 5.4.2

<JUnit>
JUnit은 자바 프로그래밍 언어용으로 작성된 오픈 소스 테스팅 프레임워크입니다. JUnit은 단위 테스트를 작성하고 실행하기 위한 도구를 제공하여 소프트웨어 개발자가 코드의 정확성을 검증할

수 있게 도와줍니다.

JUnit은 Java 언어를 기반으로 하며, Java의 다양한 개발 환경과 프레임워크에서 사용할 수 있습니다. JUnit은 개발자 커뮤니티에서 널리 사용되는 표준 테스팅 프레임워크로서, 소프트웨어 테스트의 핵심 요소 중 하나입니다.

<@RunWith, @ContextConfiguration, @Test>
@RunWith은 JUnit 테스트 실행을 커스터마이즈하는 데 사용되는 애노테이션입니다. JUnit은 기본적으로 BlockJUnit4ClassRunner를 사용하여 테스트를 실행하지만, @RunWith를 사용하여 다른 실행자를 지정할 수 있습니다. 특정 테스트 실행자를 사용하려는 경우 @RunWith를 테스트 클래스에 적용하고, 실행자 클래스를 지정해야 합니다.

@ContextConfiguration은 Spring 테스트 컨텍스트 프레임워크와 통합된 JUnit 테스트를 위해 사용되는 애노테이션입니다. 이 애노테이션은 Spring 애플리케이션 컨텍스트를 구성하고 테스트에 필요한 빈(Bean)을 로드하는 데 사용됩니다. @ContextConfiguration은 XML 파일, 자바 구성 클래스 또는 클래스 경로에 있는 구성 파일과 같은 다양한 방식으로 컨텍스트 구성을 지정할 수 있습니다.

@Test는 JUnit 프레임워크에서 단위 테스트 메서드를 식별하는 데 사용되는 애노테이션입니다. 이 애노테이션을 메서드에 적용하면 해당 메서드가 테스트 메서드임을 나타내며, JUnit은 이 메서드를 실행하여 테스트 결과를 확인합니다. @Test 애노테이션을 적용한 메서드는 테스트 실행 시 자동으로 실행되고, 테스트 결과에 따라 성공 또는 실패로 표시됩니다.

이 세 가지 애노테이션은 JUnit과 Spring 프레임워크를 함께 사용하여 테스트를 작성하고 실행하는 데 도움이 됩니다. @RunWith는 JUnit 실행자를 커스터마이즈하고, @ContextConfiguration은 Spring 컨텍스트를 구성하고 로드하며, @Test는 테스트 메서드를 식별합니다.

---

나머지 설명이 부족한 부분들이 있을수도 있습니다.
STS3 소스 코드를 참조하기 바랍니다.